Australian Government
Department of Defence
Defence Science and
Technology Organisation

HPAC (Hazard Prediction and Assessment Capability) <> jSWAT (Joint Seminar Wargaming Adjudication Tool) Integration; A Technical Solution

Matt Brennan[1] , Alex Skvortsov and Ralph Gailis

Human Protection and Performance Division
Defence Science and Technology Organisation

DSTO-TN-0721

ABSTRACT

This paper provides an outline of the technical solution to be adopted when integrating the Hazard Prediction and Assessment Capability (HPAC) with DSTO's Joint Seminar Wargaming Adjudication Tool (jSWAT). Opportunities to conduct "least path of resistance" integration between the two applications are explored to support an eventual Proof of Concept demonstration. The report concludes with some observations on achievable longer term integration goals.

[1] classForge PTY LTD

Published by

Human Protection and Performance Division
DSTO Defence Science and Technology Organisation
506 Lorimer St
Fishermans Bend, Victoria 3207 Australia

Telephone: (03) 9626 7000
Fax: (03) 9626 7999

AR-013-759
September 2006

HPAC (Hazard Prediction and Assessment Capability) <> jSWAT (Joint Seminar Wargaming Adjudication Tool) Integration; A Technical Solution

Executive Summary

The transformation of general military strategy and planning guidance to specific Defence Capability Strategy is where major future war fighting concepts are explored. The type of issues explored include Joint Operational concepts, Australian Indicative Planning Scenarios (AIPS), Force Options testing and NCW concepts using such tools and discriminators as wargaming and measures of effectiveness. Insertion of CBR scenarios or challenges into the future operating concept exploration studies would provide some tests of the robustness of concepts to realistic threats. This Technical Note investigates the integration of an existing Chemical Biological Radiological and Nuclear (CBRN) hazard prediction tool (HPAC) into the Army Experimental Framework's seminar wargaming environment (jSWAT) via a Proof of Concept (POC) demonstration.

- Ability to start/stop HPAC client and server in the background, control it from jSWAT and run it invisibly to jSWAT users. No knowledge of HPAC is required from jSWAT users.
- Ability to use full HPAC functionality in the jSWAT application through the Advanced Programming Interface (API) provided, resulting in mutual information exchange and data feed between jSWAT and HPAC functional modules.
- Ability to display HPAC data in jSWAT in industry standard format (OpenMap) .

What has been delivered:

- An environment to assist the adjudication of CBRN effects by displaying contoured data associated with an incident release. No automation is provided of casualties or manoeuvre effects as a result of exposure to a CBRN release or the need to adopt protective (MOPP) measures.
- Sensors and detectors allow the non-adjudicating players to effectively interact with the physical model of the plume. However, an adjudicator may intervene to shape the player's awareness of the source term in any manner desired.

The price and/or availability of this code (or an API that eases the task of integrating with the HPAC toolset) must be weighed in a cost/benefit analysis when considering any further development beyond POC for the HPAC<>jSWAT integration.

The HPAC<>jSWAT product is expected to provide a valuable tool for use in the Army Experimental Framework. This is timely, given the heightened importance that CBRN issues have now received in the ADF. The tool will be used to test concepts of operation in a CBRN threat environment, and guide future capability development,

both in terms of the impact of CBRN incidents on general ADF capability, as well as specific CBRN capability development and acquisition.

Contents

1. INTRODUCTION 1

2. HPAC ARCHITECTURE 2
   - 2.1 SCIPUFF Server 2
   - 2.2 HPAC Clients 3
   - 2.3 HPAC Project Editor 4
   - 2.4 The HPAC Project File 5

3. BRIDGING THE ARCHITECTURES 5
   - 3.1 The jSWAT Server as Middleware Bridge 6
   - 3.2 A User Interface to Define Incidents and Releases 7
   - 3.3 Samplers, Detectors and Effects 10

4. ARCHITECTURAL FINDINGS MAPPED TO COMPONENT DEVELOPMENT11
   - 4.1 jSWAT Server to ICE 11
   - 4.2 jSWAT Server 12
   - 4.3 jSWAT Client to Server 13
   - 4.4 jSWAT Server to Client 13
   - 4.5 jSWAT Client (1) 14
   - 4.6 Samplers (1) 14
   - 4.7 Software Release 15
   - 4.8 ICE Jar 15

APPENDIX A: DEVELOPMENT MAP AND PRIORITISATION 17

APPENDIX B: STARTING HPAC COMPONENTS IN DISTRIBUTED FASHION19
   - B.1. Introduction 19
   - B.2. Starting ICE 19
   - B.3. Starting the SCIPUFF server 20
   - B.4. Starting the HPAC Project Editor 20
   - B.5. Recovering from Failure 20

APPENDIX C: ICE EXAMPLES 23
   - C.1. Introduction 23

C.1.1 Example 1 23
C.1.2 Example 2 23
C.1.3 Example 3: 23
C.1.4 Example 4 23
C.1.5 Example 5 23
C.1.6 Example 6 23
C.1.7 Example HPAC757 23

APPENDIX D: CBR DETECTOR ICONS 25

1. Introduction

This paper canvasses a sample technical solution to rapidly integrate the Hazard Prediction and Assessment Capability (HPAC) with DSTO's Joint Seminar Wargaming Adjudication Tool (jSWAT). The intent of the process is to demonstrate a Proof of Concept (POC) level of operation and analyse the suitability of the architectures for deeper integration.

HPAC models the release of Chemical Biological Radiological and Nuclear (CBRN)materials to the atmosphere and the associated dispersion using detailed meteorological information [1]. It provides an estimate of effects on the physical environment and to a lesser extent the resulting impact of that environment on exposed population. Developed in the US by the Defence Threat Reduction Agency (DTRA), this counterproliferation / counterforce tool "assists warfighters in destroying targets containing weapons of mass destruction and responding to hazardous agent releases" [1, 4, 5]. It employs integrated source terms, high resolution weather forecasts and particulate transport algorithms to rapidly model hazard areas and human collateral effects.

The Joint Seminar Wargaming Tool is an LOD developed software package (programmed in Java) that aims to facilitate the seminar wargaming process that Army currently uses to explore new concepts. Seminar wargaming is applicable to the way in which the CBRN exercises are conducted, and the use of jSWAT enables control of scenarios and the capture of required data in an automated fashion.

jSWAT's design philosophy is based around the transparent map overlays that are used as part of the Military Appreciation Process (MAP). Different levels of tactical or strategic information are laid over the underlying terrain image to provide different insights into the planning process. jSWAT utilises this idiom in its layout, with different layers of information being able to be switched on and off at will. HPAC's CBRN information is displayed as another overlay. It provides information related to which CBRN sensors are activated and allows the operators to plot the spread of the plume

While the term "integration" sounds straightforward, it needs to be clearly specified in the current context of wargaming tools integration. On the functional level we needed to evaluate two options

- Duplex information exchange and display,
- Mutual models influence (i.e. output of one model may impact result of the other).

For the POC, the first option was adopted, i.e. HPAC data is displayed in jSWAT, but it has no direct influence on model entities. Thus as a result of models integration, we provided in jSWAT:

- An environment to assist the adjudication of CBRN effects by displaying contoured data associated with an incident release. No automation is provided of casualties or manoeuvre effects as a result of exposure to a CBRN release or the need to adopt protective (MOPP) measures.

- Sensors and detectors allow the non-adjudicating players to effectively interact with the physical model of the plume. However, an adjudicator may intervene to shape the player's awareness of the source term in any manner desired.

The integration methods may vary depending upon their level of coupling between components and the amount of internal modifications required to enable them. We also evaluated two extreme options

- Strong coupling (each model uses shared components of each other and makes direct functional calls)
- Loose coupling (each model has no detailed knowledge of the other, and only receives / sends broadcasted messages).

The second option was adopted for POC since it is less dependent on model source code and more flexible in terms of deployment and version control. This report aims to explain the details of implementation behind these broad concepts.

2. HPAC Architecture

2.1 SCIPUFF Server

At the heart of HPAC is Second-order Closure Integrated Puff (SCIPUFF), a Lagrangian puff dispersion model developed in Fortran for the US Defence Threat Reduction Agency (DTRA). Detailed descriptions of material releases, meteorology, terrain, and other inputs are fed to SCIPUFF, which calculates the transport and dispersion and tracks material concentrations, depositions, and doses.

HPAC 4.0 is designed as a networked application. The foundation of the system is a set of services designed to be accessed from a variety of clients (not only the client application built for HPAC). The HPAC 4.0 server application is exclusively available for execution on MS Windows (although various porting activities are mentioned in the HPAC design literature [1]).

Central to the application's functionality is the atmospheric dispersion engine and a layer of services wrapped around this engine. These receive detailed descriptions of releases and materials and calculate their dispersion. They also contains facilities for generating output in terms of dose and deposition contours, vertical and horizontal plume slices, tabular output, and other forms.

In addition, several models provide a more abstract or operational description of events, referred to as *incidents*. These models include conventional attacks on biological and chemical facilities, accidents at nuclear facilities, nuclear weapon incidents, nuclear weapon detonations, chem-bio weapons, missile intercepts, smoke and obscurants. Applications and systems may access HPAC functionality by interacting with HPAC services, linking with HPAC libraries, or instantiating Java HPAC components. For remote access, these services are defined using the Object Management Group (OMG) Interface

Definition Language (IDL [2]), part of the Common Object Request Broker Architecture (CORBA) specification [3].

Adhering to the notion of a Single Thread Model, HPAC servers provide a singleton factory service which is registered in the CORBA[1] naming service for discovery. The factory is responsible for providing a per-client server object to perform work on behalf of the client. These per-client instances are relatively short-lived but provide a critical limitation in terms of jSWAT integration: it is not possible for two clients to simultaneously access a running server[2]. The intention here is that clients open a session with the server, perform a calculation, request output and then release the session.

Finally, it is of note (for firewalling purposes for example), that RMI[3] is used to transport larger files from client to/from server bypassing the CORBA linkage to avoid problems with implementation fragility when passing large objects via the CORBA Object Request Broker (see reference [1] for complete details).

2.2 HPAC Clients

Client applications may be written in any language and targeted for any platform capable of communicating with the server via TCP/IP and the Common Object Request Broker Architecture (CORBA). The server is designed to support many clients communicating to the server at one of several levels including a CORBA interface. In standalone mode, the Server and the Project Editors effectively bypass the CORBA interface and communicate directly. When starting HPAC from a clean installation, this "standalone" mode is expedited by executing most server objects in the same Java Virtual Machine (JVM) as client objects, thereby bypassing CORBA object transport [4].

[1] **CORBA** is the acronym for **C**ommon **O**bject **R**equest **B**roker **A**rchitecture, OMG's open, vendor-independent architecture and infrastructure that computer applications use to work together over networks. Using the standard protocol IIOP, a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network.

[2] Although it might be fun to try.

[3] Java Remote Method Invocation (Java RMI) enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines.

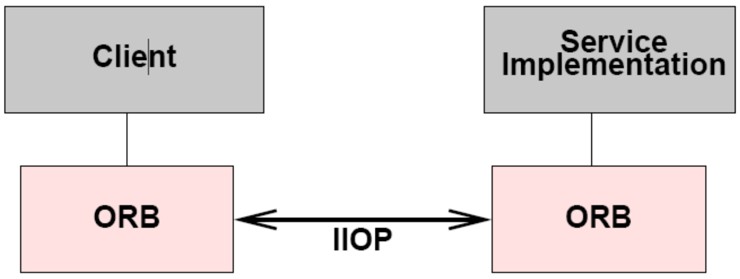


Figure 1: Client Server Communications via CORBA [5]

2.3 HPAC Project Editor

The standard Graphical User Interface (GUI) to the server is the HPAC client application known as the "*Project Editor*". Currently, OpenMap [6] is used for the bulk of map data display. Many National Imagery and Mapping Agency (NIMA) raster and vector mapping products are displayable in the Project Editor, including CADRG and CIB imagery as well as tiled data and VMAP data[4]. jSWAT shares the same capacity as the HPAC Project Editor for map display through the OpenMap toolkit allowing a common visual baseline in defining and exploring incidents and their extent of effects.

An instance of an incident is created by dragging the icon in the Incident Definition Palette onto the map Project Editor display. A typical sequence of user actions involves:

1. Creating the incident,
2. Setting the weather,
3. Calculating the effects using the Dispersion Server,
4. Display one or more output contour plots via the Output button.

A high degree of user interaction is supported. Icons representing releases may be dragged on the map to relocate them, or detailed coordinates may be entered via release edit dialogs. The map display is highly configurable, and overlays representing plumes, points of interest, and other features on the map display may be toggled on and off and order-adjusted.

The facilities and parameterisations offered via the Project Editor provide a useful reference when developing the client application as they offer a great deal of insight into the data that the server will accept for an incident, plot or export.

The complex and fully featured JavaBeans and OpenMap layers developed for the Project Editor are available for reuse in 3rd party products[4]. The existence of this capability proves the concept of being able to display HPAC outputs within the jSWAT tool which is also OpenMap based. The price and/or availability of this code (or an API that eases the task of integrating with the HPAC toolset) must be

[4] More descriptions of these data formats available at:
http://en.wikipedia.org/wiki/GIS_file_formats or from the NIMA website.

weighed in a cost/benefit analysis when considering any further development beyond POC for the HPAC<>jSWAT integration[5].

2.4 The HPAC Project File

A project is a collection of files needed for input and output by HPAC. Project files fall into two categories: ASCII (text-based) and binary (machine-readable). Generally, all project files begin with a project name followed by different extensions to indicate their purpose. A collection of project files may be "jarred" (collated and compressed in the familiar "zip" file format) into a single file to ease file management and transfer.

Project files hold the parameters (such as hazard type and environmental data) that define the hazard and store the processed outputs. For as long as project files are retained, one can reopen them, edit them, and rerun them.

The Project File is HPAC's principal mechanism to save incident/release and plot data to disk. As such, it presents one option in saving HPAC related data in association with a jSWAT scenario – and also a mechanism of transfer.

3. Bridging the Architectures

At runtime, the HPAC application can be thought of for development purposes as comprising three key elements:

- The SCIPUFF server,
- The Project Editor, and
- The Interface for Consequent Effects.

For deployment, each of these components may execute on a separate machine and are networked through the agency of a CORBA middleware layer. A NameService provides a namespace context to resolve the names of desired factory services and create instances via the ICE service "factory".

When considering jSWAT integration with HPAC, we add two elements:[7]

- The jSWAT Server, and
- One or many jSWAT Clients.

The jSWAT Server and Clients are networked via a JavaSpace [7] with Jini[6] providing a mechanism for the Clients to discover instances of the Server running on the network. The

[5] There may be an opportunity to canvas deeper access to the HPAC toolset in discussion with international stakeholders at the beginning of 2007.

[6] Jini™ (pronounced like genie) is a network architecture for the construction of distributed systems.

JavaSpace in this case functions as a middleware layer providing a simple network paradigm for sharing objects between Server and Client analogous to the CORBA middleware layer.

The jSWAT Server acts as the mediator of the JavaSpace and controls the "truth" of the active wargame. That is, the Server populates the space with objects representing the current game state and is the sole author of objects written to the space.

3.1 The jSWAT Server as Middleware Bridge

In order to integrate the two applications with their differing middleware layers, it is sufficient to create a linkage between the jSWAT Server and the ICE. This linkage acts as a bridge between the two middleware layers and suffices to translate between the object structure of HPAC on the one hand and the representation of the world in the jSWAT space on the other.

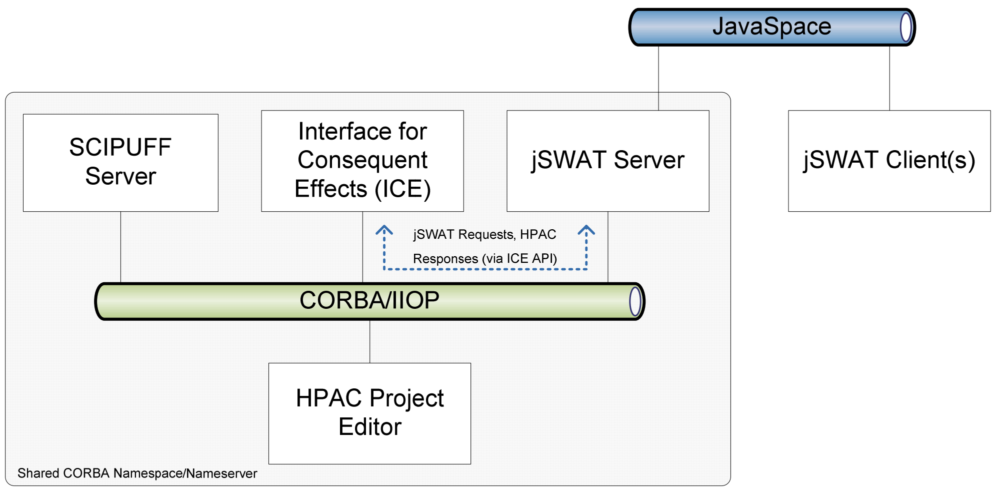

JavaSpace
SCIPUFF Server
Interface for Consequent Effects (ICE)
jSWAT Server
jSWAT Client(s)
jSWAT Requests, HPAC Responses (via ICE API)
CORBA/IIOP
HPAC Project Editor
Shared CORBA Namespace/Nameserver

Figure 2: jSWAT Server bridging JavaSpace and CORBA middleware layers

Figure 2 illustrates the role of the jSWAT server in an architectural "block diagram" view of the integrated application. Conceptually, or in fact, each of the blocks may be started in a separate Java Virtual machine on separate workstations and intercommunicate using TCP/IP (and the layered middleware services of CORBA and Jini/JavaSpaces). Figure 3 displays the top level Java classes[7] that comprise the building blocks for a widely distributed application.

[7] For display, the jSWAT package name has been truncated from "com.classforge.jswat". Similarly, the actual client and server classes instantiated at runtime are: ClientRowanWizard and ServerRowanWizard respectively.

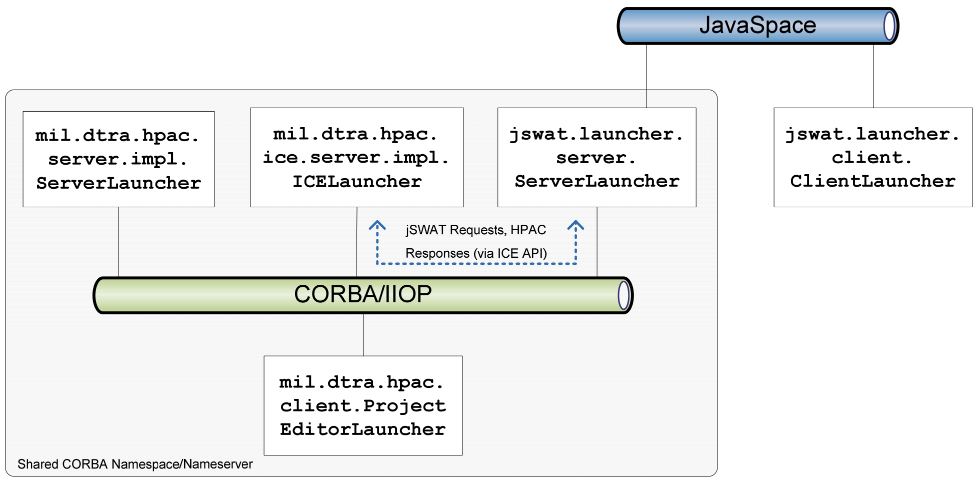

JavaSpace
mil.dtra.hpac.server.impl.ServerLauncher
mil.dtra.hpac.ice.server.impl.ICELauncher
jswat.launcher.server.ServerLauncher
jswat.launcher.client.ClientLauncher
jSWAT Requests, HPAC Responses (via ICE API)
CORBA/IIOP
mil.dtra.hpac.client.ProjectEditorLauncher
Shared CORBA Namespace/Nameserver

Figure 3: Starting jSWAT+HPAC as several Virtual Machines

3.2 A User Interface to Define Incidents and Releases

The HPAC Project Editor is a rich and fully featured user interface to create the description of some event that releases NBC material into the environment (an "incident"). HPAC cleverly characterises each incident as "Where", "What" and "When" and allows an incident to transform to a "release" (being a detailed physical description of the NBC material that is released into the environment).

In addition to defining the source term for the release, the Project Editor GUI provides the user with the ability to address a wealth of detailed information covering weather, terrain and external factors that will affect the release.

The programming effort to recreate the GUI components available in HPAC within in jSWAT is outside of the terms of the POC development. Indeed, such an effort would be entirely wasted whilst code exists for each of these panels as easily incorporated Swing[8] components[8].

For Proof of Concept then, we will rely on the Project Editor GUI to define in detail each of the incidents to be bridged to the jSWAT world. The ICE API goes so far as to suggest this approach as viable and provides the ability to copy extant incidents from a reference copy within an HPAC Project.

Because each client in the HPAC architecture does not share the server but rather is allocated its own short-lived server process, there is no capacity in the HPAC architecture

[8] Swing is Java's prime toolkit for the development of Graphical User Interfaces and underlies almost all Java client applications including the HPAC Project Editor and jSWAT.

for two clients to share a project at runtime. This key finding, mentioned in section 2.1, drives the POC integration towards the following user process using the saved Project file as the mechanism to transfer incidents from HPAC Project Editor to jSWAT Client :

1. A skilled user creates incidents with the HPAC Project Editor including source term, terrain and weather conditions.
2. The user saves the incident definition as an HPAC Project in a well known location.
3. The jSWAT server is started and pointed to the HPAC Transfer Project.
4. The Project is loaded to the SCIPUFF server using the ICE API.
5. The jSWAT JavaSpace is populated with objects corresponding to incidents for display on the jSWAT Client.
6. Periodically or at the culmination of the game, the Project file is saved to disk.

The process flow described above can be visualised (with some alternative potential pathways) per Figure 4. This figure highlights the critical issue of the separate "per client" process space for the HPAC Server and illustrates the necessity of a file based transfer between process spaces in the situation where the Project Editor is used to characterise the release.

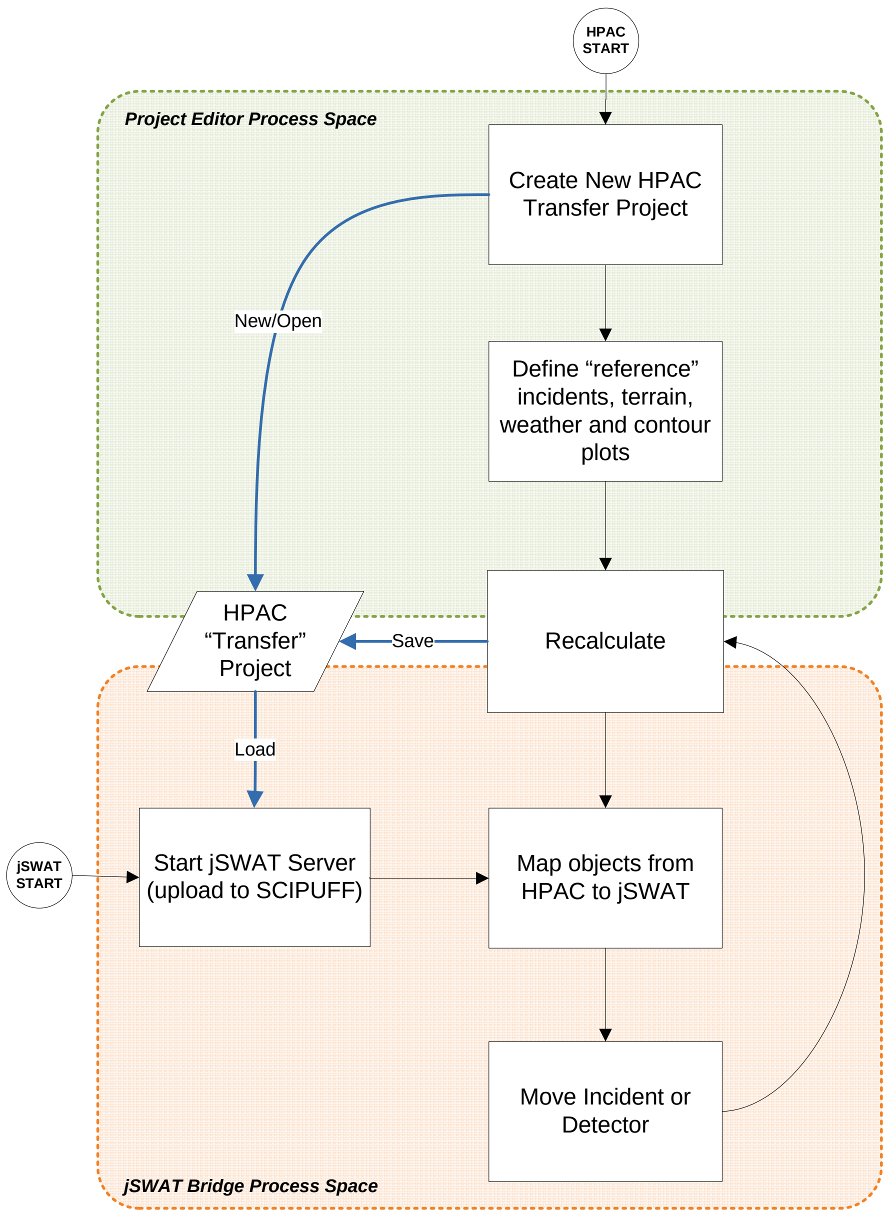

HPAC START
Project Editor Process Space
Create New HPAC Transfer Project
New/Open
Define “reference” incidents, terrain, weather and contour plots
HPAC “Transfer” Project
Save
Recalculate
Load
jSWAT START
Start jSWAT Server (upload to SCIPUFF)
Map objects from HPAC to jSWAT
Move Incident or Detector
jSWAT Bridge Process Space

Figure 4: Workflow transferring HPAC Project to jSWAT

A further refinement on this process would allow the HPAC user to save the Transfer Project during a running jSWAT game allowing automatic (or, more likely, manual) reloading and remapping of the transferred releases into the jSWAT object space. This idea, while technically feasible, falls outside the brief of the Proof of Concept demonstration. A further development that relied on the Project Editor to tweak a jSWAT mapped release might rely on the ability to:

1. load or reload the Transfer Project in the Project Editor, and
2. reload and remap a changed Transfer Project.

3.3 Samplers, Detectors and Effects

The SCIPUFF server is able to take a range of named samplers and compute (at least) concentration values as a time series at these static points. Little information has been available on the format of inputs and outputs necessitating a degree of effort to reverse engineer this process. Beyond the POC, direct contact with the developers of this element of the SCIPUFF codebase may substantially enhance our usage of the sampler interface.

The input samplers are a formatted ASCII file/string illustrated by this example of the first two lines of a sample file followed by coloured annotations:

HPAC	SENSOR	UTM	14		
preamble	preamble	UTM flag	UTM Zone		
634.476	3925.550	3.0	CONC	SF6	FRD_CBD003
easting	northing	alt (meters?)	measurement	agent	sampler name

The above example places a sulphur hexafluoride sensor in the continental US.

To place Sarin samplers on Kangaroo Island we might use the following string:

```
HPAC SENSOR  UTM 53
727.538      6019.455      10.0      CONC   GB     VAN1
727.738      6026.422      10.0      CONC   GB     VAN2
```

Once a sampler is placed(Loading a sampler file or dynamically editing information in the Project Editor), it is necessary to run the dispersion computation (a matter of seconds, minutes or more depending on the current configuration of the Project). Intuitively we might expect that these sample points will occur off the grid of points (i.e. interior to calculation cells) used by the dispersion server for calculation. Thus, an exact characterisation for any given point requires a rerun of the dispersal calculation.

While it has been possible to vary most of the fields listed above to say, change UTM zone and the agent, it has not been possible to determine whether any measurement but “CONC” (concentration) is possible.

Beyond the POC, it is conceivable that the Sampler interface be used to determine effects on personnel inside a plume. In this case some investigation might be made into other possible measurement values that may be elicited from the Sampler interface. One could conceive of using the Sampler interface to accumulate/integrate probability of casualty/mortality by MOPP level over time and reflect this directly on vulnerable game entities.

The Sampler input is further limited in being able to accept just 200 input points.

Finally, output from this process is returned as a file or string. For each sampler point a time series of Concentration, Variance (inferred) and Time Correlation (inferred)[9]. The value of concentration is assumed to be in units of mg/m^3.

4. Architectural Findings Mapped to Component Development

The findings of the previous section allow decomposition of the jSWAT application development to support the specific needs of the interface to the SCIPUFF Server via the ICE. The following section provides a few brief notes on design and architectural considerations during integration.

In general, each of the headings below refer to a development roadmap found at Appendix A: Only those items feasible for implementation during the POC Phase are enumerated below.

4.1 jSWAT Server to ICE

The interface to the ICE allows products (for example Plots) from the SCIPUFF Server to be saved as files for export. The jSWAT Server ("the Server" hereafter) will be responsible for making these exported products available to the jSWAT clients ("the Clients" hereafter) for display.

It is suggested but not necessary that the ICE instance and Server share a common filespace - a simple mechanism for this is to run both ICE and Server on the same machine (although a network file system would also suffice).

The Server will use a scratch directory within the data path of the HTTP Server sharing data content to the Clients to save and re-export file based outputs such as plots.

This directory will be of the format:

```
data/scenarios/<scenario>/output/<game name>/<run>/hpac-output/
```

[9] No information was available on the structure of the sampler files for the majority of the time period available in the POC study. Information on this became available in July 06, but this was unfortunately too late for implementation in the POC contract.

4.2 jSWAT Server

The jSWAT Server will include a Pluggable Model to dynamically include the HPAC interface at startup. In the absence of this user selection, the jSWAT application will perform as currently specified.

Where the HPAC Pluggable Model is specified for inclusion during Server startup a model object will be created and provided with the location of the NameServer by address:port. Initially this will be via a properties file with an option to have these details edited in the startup wizard.

The model will open the Transfer Project (again determined by a properties file with an option for browse and select at a later date) and, after determining a reference to the SCIPUFF Server via the ICE Factory Service, upload the transfer project into a "per client" space on the SCIPUFF Server.

All communications with the ICE will be conducted via a separate thread under timeout to prevent the condition where a failure in HPAC will cascade into a jSWAT game. Note that, in practice, the highly variable nature of dispersal duration means that this time out must be impracticably long!

Thereafter, the model will search the loaded project for Incidents and corresponding Plots. The model will maintain a mapping between HPAC Ids and jSWAT UniqueIds[10] so that changes in one or the other can be reflected either by an update to the HPAC object and recalculation or an update to the corresponding jSWAT object and publication via the GameSpace to the jSWAT clients.

The mapping table to be maintained by the Server is (roughly) described at Figure 5.

[10] Objects of the type UniqueId provide an object reference system within the jSWAT JavaSpace (GameSpace).

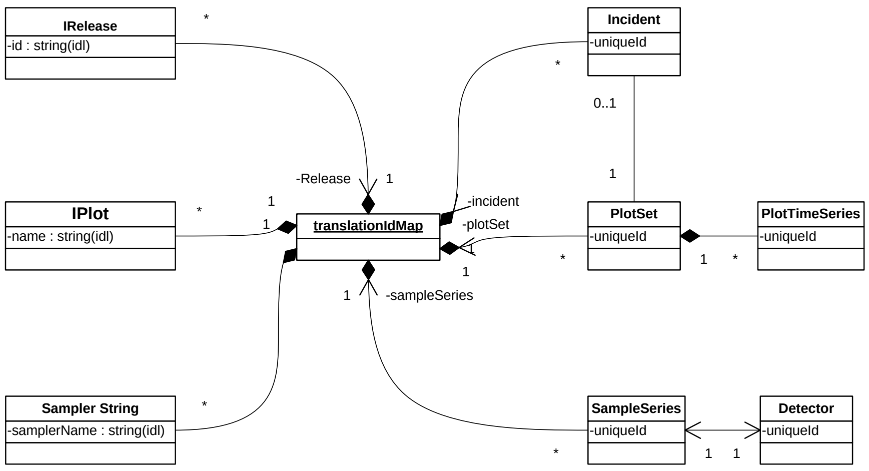


Figure 5: Translation Id Mapping maintained by jSWAT Server

4.3 jSWAT Client to Server

The jSWAT Client will be able to request the repositioning or change in start time of a given incident[11]. These operations will be accomplished via the extant jSWAT change request mechanism as a result of user action on the Client. The Server shall be responsible for detecting these changes, and queuing the need to recalculate the releases via the dispersion Server and regenerating Plots based on new time and space information.

Recalculation shall occur in a specific "recalculation" phase. That is, the Clients may make several changes to the location of Incidents and, only when a Master White Client requests will the recalculation be made.

The Client may also request new/moved/deleted detectors via the same mechanism and with the same recalculation proviso.

4.4 jSWAT Server to Client

The jSWAT Server will publish to the Client object mapped from the HPAC Transfer Project to the corresponding object time in the jSWAT GameSpace (JavaSpace).

[11] Note that in the POC implementation, it is necessary to conduct repositioning via the HPAC Project Editor.

The Server will correlate all available plots for a give release and associate them with the initiating release. The plots will be presented as a time series of viewable items where multiple plots of the same type exist for a release. The Plots will be available as shapefiles or similar for display on the jSWAT Client using the OpenMap facilities.

4.5 jSWAT Client (1)

The jSWAT client will display one Icon per Incident on the White clients (only).

The Client will allow drag and drop of an Incident to reposition the incident. A reposition event will pend recalculation of all contour data.

The Client should provide the ability to set the initiation time of a dragged incident to "time now".

The icons used to represent the Incident will be per the HPAC Project Editor to facilitate easy comparison of the tactical picture between applications (though note that the Project Editor view will "age" as location and times are changed on the jSWAT Client.

One Layer per Plot Series will be created with the ability for the user to choose (mutually exclusively) which time in the plot series will be displayed (via radio button grouping in the jSWAT Client properties panel.

The legend associated with each selected plot will be displayed in the properties panel (this may be beyond the scope of the POC effort).

4.6 Samplers (1)

The jSWAT Client will allow detectors to be placed in the environment by Blue or White players. On creating or moving a sensor, the Server will map the Sensor into a collated list for calculation of concentration time series data. This recalculation will pend until initiated by the White player.

Display on map of non zero concentration is subject to discussion (i.e.: how will the players rapidly see which Detectors have registered some concentration on a CBR agent.

The facility will be provided for White players to overrule HPAC generated concentration data to simulate false positives, failed sensors, detector jitter and meander of the plume. This situation hopes to provide a more realistic and White controlled picture to Blue (vice HPAC's integrated, averaged sensor readings).

Detector icons shall be per 0

4.7 Software Release

The jSWAT release will provide a modified batch file to start SCIPUFF/ICE separately (i.e.: not in “standalone” mode).

A modified batch file will be provided to start Project Editor in a manner that connects to the SCIPUFF server via the ICE interface (vice the normal HPAC “standalone” mode where both SCIPUFF and the Project Editor bypass the CORBA interface to communicate directly.

The HPAC icons used to illustrate releases shall be bundled in the jSWAT application. Their use in the jSWAT application is assumed to be suitable and not contravene licensing of the HPAC software.

4.8 ICE Jar

In order to call the ICE API, it will be necessary to distribute the ICE “jar” file (compressed Java class files) with the jSWAT application. Once again, this is assumed to be in accordance with the licensing of the HPAC software.

Appendix A: Development Map and Prioritisation

The following figure enumerates development tasks as part of POC development of HPAC <> jSWAT integration. Those areas marked as "(2)" are reserved for future development.

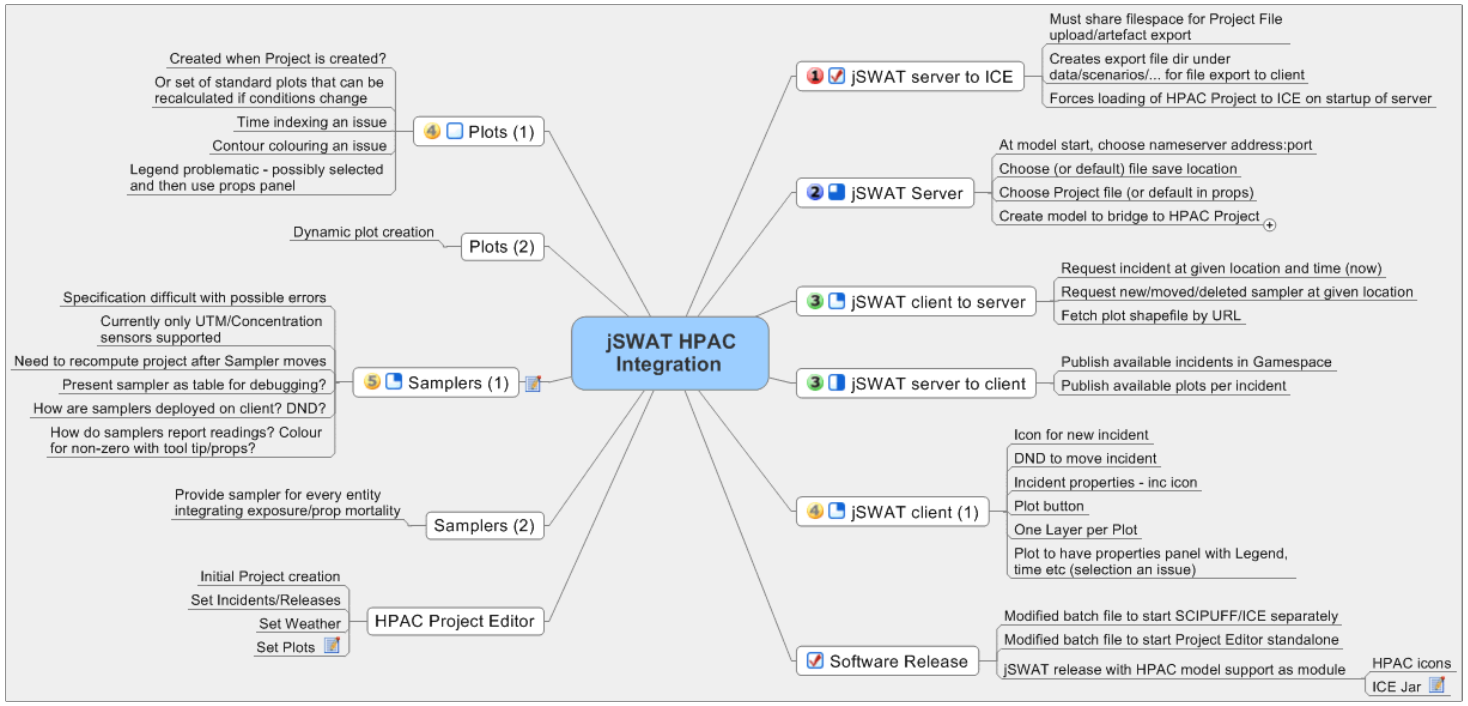

jSWAT HPAC Integration
Plots (1)
Created when Project is created?
Or set of standard plots that can be recalculated if conditions change
Time indexing an issue
Contour colouring an issue
Legend problematic - possibly selected and then use props panel
Plots (2)
Dynamic plot creation
Samplers (1)
Specification difficult with possible errors
Currently only UTM/Concentration sensors supported
Need to recompute project after Sampler moves
Present sampler as table for debugging?
How are samplers deployed on client? DND?
How do samplers report readings? Colour for non-zero with tool tip/props?
Samplers (2)
Provide sampler for every entity integrating exposure/prop mortality
HPAC Project Editor
Initial Project creation
Set Incidents/Releases
Set Weather
Set Plots
jSWAT server to ICE
Must share filespace for Project File upload/artefact export
Creates export file dir under data/scenarios/... for file export to client
Forces loading of HPAC Project to ICE on startup of server
jSWAT Server
At model start, choose nameserver address:port
Choose (or default) file save location
Choose Project file (or default in props)
Create model to bridge to HPAC Project
jSWAT client to server
Request incident at given location and time (now)
Request new/moved/deleted sampler at given location
Fetch plot shapefile by URL
jSWAT server to client
Publish available incidents in Gamespace
Publish available plots per incident
jSWAT client (1)
Icon for new incident
DND to move incident
Incident properties - inc icon
Plot button
One Layer per Plot
Plot to have properties panel with Legend, time etc (selection an issue)
Software Release
Modified batch file to start SCIPUFF/ICE separately
Modified batch file to start Project Editor standalone
jSWAT release with HPAC model support as module
HPAC icons
ICE Jar

Figure 6: Map of Development Areas and Priorities

Appendix B: Starting HPAC Components in Distributed Fashion

B.1. Introduction

This Appendix serves to record a number of lessons learned when integrating HPAC with jSWAT. The HPAC installation used was version 4.04 SP 3.

B.2. Starting ICE

The supplied ICE source code and examples were generally compatible with HPAC. However, the installation instructions applied to HPAC 4.04 without SP3 applied.

For example, the install instructions suggested that:

```
hpacserver.servers.16=\
ICEServerFactory,\
mil.dtra.hpac.ice.server.impl.ICEFactoryImpl
```

should be added to the server.properties. While this worked for "SP0", by SP3, the class structure and startup had changed such that the Factory class described no longer implemented the appropriate interface to be started. Instead, this ICE Factory seems to have been started anyway.

To correctly start ICE, the supplied batch file needed modification to at least:

```
# not start the SCIPUFF server and
set JavaProperty=%javaProperty% -Dhpac.standalone=%Standalone%

# point ICE at the naming server
set IceProperty=%IceProperty% -
Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory
set IceProperty=%IceProperty% -
Djava.naming.provider.url=iiop://%ServerHost%:%ServerPort%/
```

With these modifications, the invocation from HPAC 4.04 should be:

```
 ice ice [port] [host]
```

Where port and host are the address and port number of the naming service.

Good information on the naming service can be found at:

http://java.sun.com/j2se/1.4.2/docs/guide/idl/tnameserv.html

Since the components rely on the ORB naming service for connectivity, ICE, SCIPUFF and any jSWAT components reliant on the naming service must thus be started after the naming server.

B.3. Starting the SCIPUFF server

Starting the SCIPUFF server with the supplied ICE batchfile implies starting the CORBA naming service as well. For correct operation, it seemed most robust to stop any extant name servers before starting the SCIPUFF and nameserver (though this wasn't always necessary – and shouldn't be required).

The ice.bat file supplied needed no modifications and should be invoked as:

```
ice scipuff [port]
```

This starts the services on localhost with the nameserver at the given port.

B.4. Starting the HPAC Project Editor

Once the SCIPUFF server and ICE are running, it is possible to connect the HPAC Project Editor to the SCIPUFF instance using a modified batch file to ensure that the SCIPUFF server is not started with the Project Editor.

The following should be considered in starting:

```
# Don't start the SCIPUFF server.
set JavaProperty=%javaProperty% -Dhpac.standalone=false

# Look for it via the ORB.
set JavaProperty=%javaProperty% -
Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory
set JavaProperty=%javaProperty% -
Djava.naming.provider.url=iiop://%ServerHost%:%ServerPort%/
```

B.5. Recovering from Failure

On occasions, applications communicating with the ICE generate exceptions that either cause or reflect that fact that the ICE has failed. A restart of ICE is required – although no other components should need a restart.

By changing the output files when starting ICE to:

```
set JavaProperty=%javaProperty% -Dhpac.stderr="stderr-ice.txt"
set JavaProperty=%javaProperty% -Dhpac.stdout="stdout-ice.txt"
```

A log separate to the SCIPUFF server is created.

Similarly, on occasion the SCIPUFF server fails. A separate log should be created vice:

```
   set JavaProperty=%javaProperty% -Dhpac.stderr="stderr-scipuff.txt"
   set JavaProperty=%javaProperty% -Dhpac.stdout="stdout-scipuff.txt"
```

allowing for post crash investigation. The SCIPUFF server can be restarted safely but obviously any per-client session information will be lost and the project must be reloaded.

The SCIPUFF server can also be quite hungry for memory when supporting multiple clients. The HPAC_Getting_Started.pdf (distributed with the HPAC application) has good advice about this although the startup batch files are ambiguous in how they apply the properties to the started Virtual Machines.

Appendix C: ICE Examples

C.1. Introduction

This appendix serves as a brief reference to the functionality in the provided ICE examples. The examples are a patchy, poorly commented set relying on some unsupplied files. They provide some insight into the modes of calling the ICE with a little supplemental documentation to guess at the expected results. In the absence of other information, they are invaluable as a source of the basic calling patterns required for talking to the ICE server[12].

C.1.1 Example 1

- Creates a radiological weapon and fetches a variety of properties.
- Demonstrates retrieval of contour files and such.
- Historical weather

C.1.2 Example 2

- Generates an Anthrax incident from the Analytical factory with continuous release.

C.1.3 Example 3

- Demonstrates use of the MissileIntercept factory and calculates Probability of Mortality for an Anthrax payload.

C.1.4 Example 4

- Nuclear weapon with fixed weather.
- Demonstrates the difficulty in having to query the server for available choices before setting one vice referencing an extant constant.
- Variety of plots including casualties and structures.

C.1.5 Example 5

- Nuclear incident.

C.1.6 Example 6

- Radiological weapon.

C.1.7 Example HPAC757

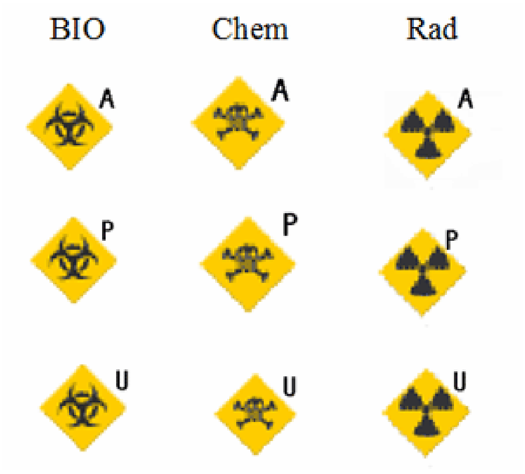
- Nuclear weapon from strike database.
- Export of tab delimited tables.

Formatted: Bullets and Numbering

[12] Note that server.shutdown() kills the ICE Server and should be commented out where it appears.

CBR Detector Icons

The following CBR Detector symbols were provided by Major David Bergman SO2 Survivability and Sustainability, FDG:



Where:

```
A == Area
P == Point
U == UAV
```

Bibliography

1. Lee, R.W., *Moving the Hazard Prediction and Assessment Capability to a Distributed, Portable Architecture*, in *Oak Ridge National Laboratory*. 2002.
2. Object Management Group Inc., *IDL to Java Language Mapping Specification (formal, 99-07-03)*. 1999.
3. Object Management Group Inc., *The Common Object Request Broker: Architecture and Specification, Revision 2.3*. 1999.
4. *HPAC 4.0 Architecture Reengineering* 2003.
5. Defence Threat Reduction Agency (DTRA), *Interface Control Document (ICD) For the Hazard Prediction and Assessment Capability (HPAC) Version 4.0.1*. 2002.
6. BBN Technologies. *OpenMap Open Systems Mapping Technology*. [cited; Available from: http://openmap.bbn.com.
7. Sun Microsystems. *Jini Network Technology*. [cited; Available from: http://www.sun.com/software/jini/.
8. Defence Threat Reduction Agency (DTRA), *HPAC Java Beans and Components: Application Programming Interface and Description*. 2002.

DISTRIBUTION LIST

"As per the Research Library's *Policy on electronic distribution of official series reports* (http://web-vic.dsto.defence.gov.au/workareas/library/aboutrl/roles&policies/mission.htm) Unclassified (both Public Release and Limited), xxx-in-confidence and Restricted reports and their document data sheets will be sent by email through DRN to all recipients with Australian defence email accounts who are on the distribution list apart from the author(s) and the task sponsor(s). Other addressees and Libraries and Archives will also receive hardcopies."

HPAC (Hazard Prediction and Assessment Capability) <> jSWAT (Joint Seminar Wargaming Adjudication Tool) Integration; A Technical Solution

Matt Brennan , Alex Skvortsov and Ralph Gailis

AUSTRALIA

DEFENCE ORGANISATION	**No. of copies**
Task Sponsor	
COMD LWDC	1 Printed
S&T Program	
Chief Defence Scientist	1
Deputy Chief Defence Scientist Policy	1
AS Science Corporate Management	1
Director General Science Policy Development	1
Counsellor Defence Science, London	Doc Data Sheet
Counsellor Defence Science, Washington	Doc Data Sheet
Scientific Adviser to MRDC, Thailand	Doc Data Sheet
Scientific Adviser Joint	Doc Data Sht & Dist List
Navy Scientific Adviser	Doc Data Sht & Dist List
Scientific Adviser – Army	Doc Data Sht & Dist List
Air Force Scientific Adviser	Doc Data Sht & Dist List
Scientific Adviser to the DMO	Doc Data Sht & Dist List
Chief of HPPD	Doc Data Sht & Dist List
Research Leader Chris Woodruff	Doc Data Sht & Dist List
Research Leader (HPPD), Nick Beagley	Doc Data Sht & Dist List
Research Leader (AOD) , Bruce Fairlie,	1 Printed
Research Leader (LOD) , Neville Curtis, ,	1 Printed
Research Leader (MOD) , Alan Theobald,	1 Printed
Head CBR Hazard Management (HPPD), Ralph Leslie,	1 Printed
Head Operations Research Capability (AOD), Simon Goss ,	1 Printed
Head Task Force Modernisation (LOD), Dean Bowley,	1 Printed

Head Littoral Operational Command Support (MOD), Dan Conley,	1 Printed
Task Manager (HPPD), Ralph Gailis,	1 Printed
L11 FRAC Coordinator David Bergman,	1 Printed
DNCWI,	1 Printed
Author(s): *Matt Brennan, Alex Skvortsov and Ralph Gailis*	3 Printed
DSTO Library and Archives	
Library Fishermans Bend	Doc Data Sheet
Library Edinburgh	1 printed
Defence Archives	1 printed
Capability Development Group	
Director General Maritime Development	Doc Data Sheet
Director General Capability and Plans	Doc Data Sheet
Assistant Secretary Investment Analysis	Doc Data Sheet
Director Capability Plans and Programming	Doc Data Sheet
Chief Information Officer Group	
Head Information Capability Management Division	Doc Data Sheet
Director General Australian Defence Simulation Office	Doc Data Sheet
AS Information Strategy and Futures	Doc Data Sheet
Director General Information Services	Doc Data Sheet
Strategy Group	
Assistant Secretary Strategic Planning	Doc Data Sheet
Assistant Secretary International and Domestic Security Policy	Doc Data Sheet
Navy	
Maritime Operational Analysis Centre, Building 89/90 Garden Island Sydney NSW	Doc Data Sht & Dist List
Deputy Director (Operations)	
Deputy Director (Analysis)	
Director General Navy Capability, Performance and Plans, Navy Headquarters	Doc Data Sheet
Director General Navy Strategic Policy and Futures, Navy Headquarters	Doc Data Sheet
Air Force	
SO (Science) - Headquarters Air Combat Group, RAAF Base, Williamtown NSW 2314	Doc Data Sht & Exec Summary
Staff Officer Science Surveillance and Response Group	Doc Data Sht & Exec Summary
Army	
ABCA National Standardisation Officer Land Warfare Development Sector, Puckapunyal	Doc Data Sheet

J86 (TCS GROUP), DJFHQ	Doc Data Sheet
SO (Science) - Land Headquarters (LHQ), Victoria Barracks NSW	Doc Data Sht & Exec Summary
SO (Science) - Special Operations Command (SOCOMD), R5-SB-15, Russell Offices Canberra	Doc Data Sht & Exec Summary
SO (Science), Deployable Joint Force Headquarters (DJFHQ) (L), Enoggera QLD	Doc Data Sheet

Joint Operations Command

Director General Joint Operations	Doc Data Sheet
Chief of Staff Headquarters Joint Operations Command	Doc Data Sheet
Commandant ADF Warfare Centre	Doc Data Sheet
Director General Strategic Logistics	Doc Data Sheet

Intelligence and Security Group

AS Concepts, Capability and Resources	1
DGSTA , Defence Intelligence Organisation	1
Manager, Information Centre, Defence Intelligence Organisation	1
Director Advanced Capabilities	Doc Data Sheet

Defence Materiel Organisation

Deputy CEO	Doc Data Sheet
Head Aerospace Systems Division	Doc Data Sheet
Head Maritime Systems Division	Doc Data Sheet
Program Manager Air Warfare Destroyer	Doc Data Sheet
Guided Weapon & Explosive Ordnance Branch (GWEO)	Doc Data Sheet
CDR Joint Logistics Command	Doc Data Sheet

OTHER ORGANISATIONS

National Library of Australia	1
NASA (Canberra)	1

UNIVERSITIES AND COLLEGES
Australian Defence Force Academy

Library	1
Head of Aerospace and Mechanical Engineering	1
Hargrave Library, Monash University	Doc Data Sheet

OUTSIDE AUSTRALIA

INTERNATIONAL DEFENCE INFORMATION CENTRES

US Defense Technical Information Center	1
UK Dstl Knowledge Services	1
Canada Defence Research Directorate R&D Knowledge & Information Management (DRDKIM)	1
NZ Defence Information Centre	1

ABSTRACTING AND INFORMATION ORGANISATIONS

Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, US	1
Materials Information, Cambridge Scientific Abstracts, US	1
Documents Librarian, The Center for Research Libraries, US	1
SPARES	5 Printed

Total number of copies: 39 Printed: 20 PDF: 19

Page classification: UNCLASSIFIED

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. PRIVACY MARKING/CAVEAT (OF DOCUMENT) ,

2. TITLE HPAC (Hazard Prediction and Assessment Capability) <> jSWAT (Joint Seminar Wargaming Adjudication Tool) Integration; A Technical Solution	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION) Document (U) Title (U) Abstract (U)"
4. AUTHOR(S) Matt Brennan , Alex Skvortsov and Ralph Gailis	5. CORPORATE AUTHOR DSTO Defence Science and Technology Organisation 506 Lorimer St Fishermans Bend Victoria 3207 Australia

6a. DSTO NUMBER DSTO-TN-0721	6b. AR NUMBER AR-013-759	6c. TYPE OF REPORT Technical Note	7. DOCUMENT DATE September 2006

8. FILE NUMBER 2006/1137153/1	9. TASK NUMBER 05/181	10. TASK SPONSOR COMD LWDC	11. NO. OF PAGES 26	12. NO. OF REFERENCES 8

13. URL on the World Wide Web http://www.dsto.defence.gov.au/corporate/reports/DSTO-TN-0721.pdf	14. RELEASE AUTHORITY Chief, Human Protection and Performance Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DSTO RESEARCH LIBRARY THESAURUS http://web-vic.dsto.defence.gov.au/workareas/library/resources/dsto_thesaurus.htm

HPAC, jSWAT Integration

19. ABSTRACT

This paper provides an outline of the technical solution to be adopted when integrating the Hazard Prediction and Assessment Capability (HPAC) with DSTO's Joint Seminar Wargaming Adjudication Tool (jSWAT). Opportunities to conduct "least path of resistance" integration between the two applications are explored to support an eventual Proof of Concept demonstration. The report concludes with some observations on achievable longer term integration goals.

Page classification: UNCLASSIFIED